Differentially Private Matrix Factorization using Sketching Techniques

Raghavendran Balu
Inria Rennes
Bretagne-Atlantique, France
raghavendran.balu@inria.fr

Teddy Furon
Inria Rennes
Bretagne-Atlantique, France
teddy.furon@inria.fr

ABSTRACT

Collaborative filtering is a popular technique for recommendation system due to its domain independence and reliance on user behavior data alone. But the possibility of identification of users based on these personal data raise privacy concerns. Differential privacy aims to minimize these identification risks by adding controlled noise with known characteristics. The addition of noise impacts the utility of the system and does not add any other value to the system other than enhanced privacy. We propose using sketching techniques to implicitly provide the differential privacy guarantees by taking advantage of the inherent randomness of the data structure. In particular, we use count sketch as a storage model for matrix factorization, one of the successful collaborative filtering techniques. Our model is also compact and scales well with data, making it well suitable for large scale applications.

Keywords

Count sketch, differential privacy, matrix factorization, recommender system, collaborative filtering

1. INTRODUCTION

Recommendation systems has become an indispensable component in Internet based services. It helps user in discovering new products and services by getting personalized suggestions based on their past consumption. Collaborative filtering is one of the popular techniques used in recommendation system. It uses user-item ratings relationship to provide recommendations. Relying on user behavior alone makes it domain independent and requires minimal external knowledge. Many techniques have been suggested in the past including the popular *latent factor* model. It maps both users and items to a low dimensional representation, retaining pairwise similarity. Matrix factorization learns these representation vectors from some observed ratings. They are then used to predict (by inner product) the missing entries and thereby to fill the incomplete user-item matrix [18].

There is a cost to it and that is user privacy. The consumed items have to be collected at a centralized database, to perform both analysis and prediction, which compromises user privacy. The usual anonymization alone is not sufficient as demonstrated in [4]. This calls for robust privacy preserving techniques. One such is differential privacy [10], which has gained wide out reach and acceptance from the academic community. Differential privacy provides strong theoretical guarantees and is robust to auxiliary information. There are various mechanisms that can provide differential privacy like Laplacian mechanism [10], exponential mechanism [21] and Bayesian inference [28].

Our main idea is to promote a sketching technique for matrix factorization to design a highly scalable recommendation system. The prize to be paid is a loss of accuracy in the recommendations because the sketch inherently induces noise. Yet, this noise has two benefits (which are indeed related): it provides a better regularization and protects privacy. We motivate this last argument by stressing the similarities with recent Bayesian learning enhancing differential privacy. We propose an experimental protocol to measure privacy from predicted ratings to validate our claims.

Section 2 introduces the scientific background relevant to this paper. Section 3 describes our differentially private factorization algorithm and Section 4 outlines its main benefits. In Section 5 shows the experimental assessment of the claimed advantages.

2. BACKGROUND

2.1 Matrix factorization

In a typical collaborative filtering system, the given data is a sparse matrix $\mathbf{R}$ with non zero entries $r_{u,i}$ representing the rating provided by the user $u \in \mathbb{U}$ for a given item $i \in \mathbb{I}$. Each row vector in $\mathbf{R}$ corresponds to an user u and column vector to an item i. In latent factor models, each user u is associated with a vector $\mathbf{p}_u \in \mathbb{R}^d$. Similarly, item i is associated to a vector $\mathbf{q}_i \in \mathbb{R}^d$. The goal is to approximate the rating $r_{u,i}$ by a simple scalar product $\mathbf{p}_u^\top \mathbf{q}_i$. The latent vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ of all users and items are represented as $d \times |\mathbb{U}|$ matrix $\mathbf{P}$ and $d \times |\mathbb{I}|$ matrix $\mathbf{Q}$. We need to find the latent factors from the observed ratings such that the missing entries of $\mathbf{R}$ can be predicted by approximating matrix $\mathbf{R} \approx \mathbf{P}^\top \mathbf{Q}$. This is done by minimizing the functional:

$$\mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q}) = \sum_{\text{observed } u,i} \mathcal{L}(r_{u,i}, \mathbf{p}_u^\top \mathbf{q}_i) + \frac{\lambda}{2}(\|\mathbf{p}_u\|^2 + \|\mathbf{q}_i\|^2). \tag{1}$$

$\mathcal{L}(r_{u,i}, \mathbf{p}_u^\top \mathbf{q}_i)$ quantifies the error between an observed rating $r_{u,i}$ and its prediction $\hat{r}_{u,i} = \mathbf{p}_u^\top \mathbf{q}_i$. In the sequel, $\mathcal{L}(r_{u,i}, \mathbf{p}_u^\top \mathbf{q}_i) = (r_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i)^2/2$. The second term is a penalty preventing overfitting the latent factors to the training data (*i.e.* the observed ratings) for a better generalization over the missing entries of $\mathbf{R}$.

2.2 Finding latent factors

Techniques to efficiently find the latent factors $(\mathbf{P}, \mathbf{Q})$ minimizers of the objective function $\mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q})$ of (1) work either offline or online. Offline techniques such as alternate least squares, gradient descent or online approaches such as the stochastic gradient descent proved to work well. This paper considers online techniques as its goal is to cope with the dynamicity of real-world recommendation systems. It is thus insightful to describe the approach proposed in [18].

At each step, the stochastic gradient descent randomly picks an observed rating $r_{u,i}$ and optimizes the parameters with respect to that rating. This update relies on the gradient of the loss function with respect to parameters, controlled by the learning rate η:

$$\mathbf{p}_u \leftarrow \mathbf{p}_u - \eta \nabla_u \mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q}), \quad (2)$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i - \eta \nabla_i \mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q}), \quad (3)$$

where $\nabla_u \mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q})$ denotes the gradient of the functional w.r.t. latent vector $\mathbf{p}_u$:

$$\nabla_u \mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q}) = \frac{\partial \mathcal{L}(r_{u,i}, \hat{r}_{u,i})}{\partial \hat{r}_{u,i}} \frac{\partial \hat{r}_{u,i}}{\partial \mathbf{p}_u} + \lambda \mathbf{p}_u \quad (4)$$

$$= (r_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i)\mathbf{q}_i + \lambda \mathbf{p}_u. \quad (5)$$

In the same way, $\nabla_i \mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q})$ denotes the gradient of the functional w.r.t. latent vector $\mathbf{q}_i$.

As the algorithm is sequential, only the latent factors have to be stored in main memory. The updates can be parallelized as they are local only to the parameters corresponding to a particular rating. Hence the algorithm can scale well to increasing data size.

However, the number of latent factors is linear with the number of users and items. At very large scale, when the number of users and/or items becomes extremely large (millions), then the resulting memory consumption becomes problematic. Furthermore, determining the latent factors is challenging in a dynamic environment, particularly when new users and items are added every now and then. Allocating a d-dimensional vector to every sporadically recurring new user or item quickly becomes non-tractable.

A more compact representation for the factors is therefore needed. We propose the adoption of count sketches that are typically used in other contexts. We therefore review sketching techniques before discussing their ability to fit within a recommendation context.

2.3 Sketching techniques

Sketching is an active area of development, particularly in a streaming setup. A data structure maintaining a particular synopsis of the data irrespective of the history of updates can be called a sketch [7]. Sketching technique have been applied for estimating the items frequency [8], finding similar items [12] and also limited numerical linear algebra operation [30]. The popularity of sketching techniques is attributed to its runtime and space efficiencies. We are in particular interested in the count sketch [5]. It was originally proposed to find heavy hitters, but can also be used to approximate the frequencies in turnstile model.

Count sketch is a probabilistic data structure originally designed to maintain approximations of quantities constantly updated in a datastream, but with sub-linear space complexity. Define $[N] := \{1, \cdots, N\}$ and $\{q_1^{(t)}, \cdots, q_N^{(t)}\}$ N quantities. We observe a datastream of quantity updates: $< \cdots \delta q_e, \cdots, \delta q_\ell, \cdots >$ and we would like to monitor the quantities over the time: $q_e^{(t+1)} = q_e^{(t)} + \delta q_e$.

A count sketch is represented by a $k \times w$ matrix $\mathbf{C}$ and two sets of *pairwise independent* hash functions $\{h_j(\cdot), s_j(\cdot)\}_{j=1}^k$. The *address* hash function $h_j(\cdot)$ maps an element of $[N]$ to the set $\{1, ..., w\}$ and the *sign* hash function s_j maps an element of $[N]$ to $\{+1, -1\}$. There are two typical processes: the update and the query actions.

Update: Upon the reception of the update δq_e of the e-th quantity, k entries of matrix $\mathbf{C}$ are updated: $\forall j, 1 \leq j \leq k$

$$c_{j,h_j(e)} \leftarrow c_{j,h_j(e)} + s_j(e) \cdot \delta q_e. \quad (6)$$

Query: At time t, given a query index e, mean or median of $\{s_j(e) c_{j,h_j(e)}\}_{j=1}^k$ is returned as an approximation of $q_e^{(t)}$. The median operator is more robust to noise [5] but the mean is easier to compute. In the sequel, we choose the mean operator: $\tilde{q}_e^{(t)} = k^{-1} \sum_{j=1}^k s_j(e) c_{j,h_j(e)}$.

The accuracy of the estimation is related to the size of the count sketch [7]. Note that if we query index e just before and after its update, the difference of the approximates is the true update: $\tilde{q}_e^{(t+1)} - \tilde{q}_e^{(t)} = \delta q_e$. However, updating the e-th quantity might have modified the others due to collision. In the j-th row of the count sketch, one entry has been modified by $\pm \delta q_e$ (see (6)) whereas the $w - 1$ others remained the same. In other words, the entries of $\mathbf{C}$ have been modified by random variables i.i.d. according to the p.m.f. $(1 - w^{-1})\partial_0 + (2w)^{-1}(\partial_{\delta q_e} + \partial_{-\delta q_e})$, where ∂_a represent the Dirac distribution on $x = a$. The expectation is zero and the variance $(\delta q_e)^2/w$. This implies that the update of the e-th quantity adds on all the others $\tilde{q}_{e'}$, $e' \neq e$, a centered noise of variance $(\delta q_e)^2/wk$.

Overall, the estimate based on the mean operator is unbiased with variance σ^2/wk, where $\sigma^2 = \sum_{e \in [N]} (\delta q_e)^2$. For a given (w, k), the accuracy decreases with N because the variance of the estimate increases with σ^2. In other words, the representational capacity N of count sketch can be controlled by varying (w, k).

2.4 Differential privacy

Differential privacy argues that absolute privacy is impossible and instead settles to relative level. It has garnered wide spread attention among the research community for its theoretical rigorousness and robustness to side information.

2.4.1 *In the context of recommendation system*

In our application, differential privacy convinces users to submit their ratings by showing that an attacker querying the recommendation system has difficulty in deciding whether a particular rating has been used for learning the latent factors. Even with the side-information that user u rates item i by the true value $r_{u,i}$, the attacker cannot say whether the user submitted or not this information. We are thus interested at ϵ-DP at the rating level with a trusted recommendation system.

Denote by $\mathcal{D}$ the dataset of observed ratings used for learn-

ing the latent vectors, and $\mathcal{D}' = \mathcal{D} \cup \{< u', i', r_{u',i'} >\}$ s.t. these two datasets differs by one rating. Denote by $\hat{r}_{u,i}(\mathcal{D})$ the output of a recommendation system trained on dataset $\mathcal{D}$ and queried about user u and item i. An ϵ-DP recommendation system satisfies: $\forall (a,b) \in \mathbf{R}^2, a < b, \forall u \in \mathbb{U}, \forall i \in \mathbb{I}$

$$\Pr[a < \hat{r}_{u,i}(\mathcal{D}) < b] \leq e^{\epsilon} \Pr[a < \hat{r}_{u,i}(\mathcal{D}') < b]. \quad (7)$$

2.4.2 A posteriori sampling

Paper [28] recently proved that Bayesian posterior sampling is differentially private to some extent. We explain it as follows in the context of matrix factorization based recommendation system. We first need a Bayesian framework. We assume the following prior distribution of latent factors:

$$p(\mathbf{P}, \mathbf{Q}) = \Pi_{u=1}^{\mathbb{U}} p(\mathbf{p}_u) \Pi_{i=1}^{\mathbb{I}} p(\mathbf{q}_i), \quad (8)$$

with $p(\mathbf{p}_u)$ and $p(\mathbf{q}_i)$ are Gaussian distribution $\mathcal{N}(\mathbf{0}_d, \lambda^{-1}\mathbf{I}_d)$ with $\mathbf{0}_d$ the all zero $d \times 1$ vector, $\mathbf{I}_d$ the identity matrix of size d and $\lambda > 0$. We assume a conditional pdf of the rating knowing the latent factors (*i.e.* the likelihood):

$$p(r_{u,i}|\mathbf{P}, \mathbf{Q}) \cong \mathcal{N}(\mathbf{p}_u^\top \mathbf{q}_i, \nu^2), \text{ with } \nu = 1. \quad (9)$$

This implies that, once some ratings are observed, the a posteriori distribution of the latent factors is

$$p(\mathbf{P}, \mathbf{Q}|_{\text{observed}} < u, i, r_{u,i} >) \propto e^{-\mathcal{R}_\lambda(\mathbf{P},\mathbf{Q})}, \quad (10)$$

with $\mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q})$ defined in (1). Therefore, minimizing this functional as proposed in Sec. 2.2 amounts to chose $(\mathbf{P}, \mathbf{Q})$ as their MAP (Maximum A Posteriori) estimates.

Instead of doing this, [28, 19] shows that drawing $(\mathbf{P}, \mathbf{Q})$ according to their a posteriori distribution (10) enables differential privacy: if $|\log p(r_{u,i}|\mathbf{P}, \mathbf{Q})| < B$, then the system is ϵ-DP with $\epsilon = 2B$. If $2B$ is too big, then one may scale down $\log p(r_{u,i}|\mathbf{P}, \mathbf{Q})$, which amounts to pick $\nu > 1$, *i.e.* a smoother conditional distribution.

The final issue is how to draw according to such a complex a posteriori distribution. Recent papers show that perturbating the stochastic gradient descent by some Gaussian noise is an efficient way to simulate such a sampling [29, 27]. This technique is called Stochastic Gradient Langevin Dynamics (SGLD). In our context, this would mean replacing (2) and (3) by:

$$\mathbf{p}_u \leftarrow \mathbf{p}_u - \eta \nabla_u \mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q}) + \sqrt{\eta} B_P, \quad (11)$$

$$\mathbf{q}_i \leftarrow \mathbf{q}_i - \eta \nabla_i \mathcal{R}_\lambda(\mathbf{P}, \mathbf{Q}) + \sqrt{\eta} B_Q, \quad (12)$$

with B_P and $B_Q \sim \mathcal{N}(0, 1)$.

3. COUNT SKETCHING LARGE MATRIX FACTORIZATION

3.1 Sketching vectors

In regular matrix factorization, d-dimensional latent vectors $\{\mathbf{p}_u\}_{u \in \mathbb{U}}$ and $\{\mathbf{q}_i\}_{i \in \mathbb{I}}$ are stored as dense arrays $\mathbf{P}$ and $\mathbf{Q}$, contiguous in memory. This facilitates indexing on the two dimensional array by increments of d. We propose replacing this matrix representation with a single count sketch. Although user and item vectors carry different semantic, their underlying representations are the same. Therefore we store both of them in the same structure, which should provide estimates for $N = d(|\mathbb{U}| + |\mathbb{I}|)$ elements. For the sake of clarity, we introduce two families of address hash functions: $\{h_j^u(\cdot)\}_{j=1}^k$ for the users, $\{h_j^i(\cdot)\}_{j=1}^k$ for the items. Same for the sign hash functions.

Varying (w, k) explores the trade-off between the storage efficiency and the quality of the estimation. The storage improvement comes at the cost of increasing the retrieval complexity from $O(d)$ to $O(kd)$ for a d-dimensional vector.

3.2 Sketch based factorization

The sketch based online factorization differs from regular online factorization (Sec. 2) in the latent factor queries and gradient updates merging. When a new tuple $< u, i, r_{u,i} >$ arrives, the count sketch is first queried to approximately reconstruct user and item latent vectors. Both user ID u and component index l, $1 \leq l \leq d$, are used as inputs to the k pairs of address and sign hash functions to get a mean estimate of the vector component (the same holds for item):

$$\tilde{p}_{u,l} = \frac{1}{k} \sum_{j=1}^{k} s_j^u(u, l) \cdot c_{j,h_j^u(u,l)}, \quad \forall l \in \{1, \cdots, d\}, \quad (13)$$

$$\tilde{q}_{i,l} = \frac{1}{k} \sum_{j=1}^{k} s_j^i(i, l) \cdot c_{j,h_j^i(i,l)}, \quad \forall l \in \{1, \cdots, d\}. \quad (14)$$

The estimated rating $\hat{r}_{u,i} = \tilde{\mathbf{p}}_u^\top \tilde{\mathbf{q}}_i$ is compared with the observed $r_{u,i}$ to get the loss $\mathcal{L}(r_{u,i}, \hat{r}_{u,i})$.

If $\mathcal{L}(r_{u,i}, \hat{r}_{u,i}) \leq \epsilon$, the gradient updates for $\tilde{\mathbf{p}}_u$ and $\tilde{\mathbf{q}}_i$ are just computed as in (2) and (3). Then for each component of $\tilde{\mathbf{p}}_u$ (as well as $\tilde{\mathbf{q}}_i$), the k respective cells $\mathbf{C}$ are updated with their sign corrected gradients:

$$c_{j,h_j^u(u,l)} \leftarrow c_{j,h_j^u(u,l)} - \eta s_j^u(u, l) \nabla_{u,l} \mathcal{R}_\lambda(\tilde{\mathbf{P}}, \tilde{\mathbf{Q}}), \quad (15)$$

$$c_{j,h_j^i(i,l)} \leftarrow c_{j,h_j^i(i,l)} - \eta s_j^i(i, l) \nabla_{i,l} \mathcal{R}_\lambda(\tilde{\mathbf{P}}, \tilde{\mathbf{Q}}). \quad (16)$$

$\nabla_{u,l} \mathcal{R}_\lambda(\tilde{\mathbf{P}}, \tilde{\mathbf{Q}})$ denotes the l-th component of the gradient defined in (5), and similarly for the item gradient.

Otherwise (*i.e.* $\mathcal{L}(r_{u,i}, \hat{r}_{u,i}) > \epsilon$), the cells are not updated at all. This is equivalent to not taking into account this particular observation.

4. BENEFITS OF OUR APPROACH

4.1 Space gain

The use of the count sketch is motivated as a means to trade space versus complexity. The count sketch has a storage space of wk scalars whereas the regular factorization would need $(|\mathbb{U}| + |\mathbb{I}|)d$ scalars to store the latent factors. We define the space gain γ as

$$\gamma := \frac{(|\mathbb{U}| + |\mathbb{I}|)d}{wk} \geq 1. \quad (17)$$

On the other hand, at each update triggered by observation $< u, i, r_{u,i} >$, the count sketch adds $O(2dk)$ more operations to query and update the d components of $\mathbf{p}_u$ and $\mathbf{q}_i$.

4.2 Self regularization

The Tikhonov regularization (*i.e.* L_2 norm based) is generally associated with matrix factorization, as it can be controlled by λ and customized well to the needs. There are other methods to regularize like corrupting the input data. It is shown by Bishop et al in [3] that training with corrupted data is equivalent to Tikhonov regularization.

In our scheme, we observe that the sketch structure itself regularizes the learnt latent vectors. At reception of rating $r_{u,i}$, the gradient of the regular factorization is computed based on the error $(r_{u,i}-\mathbf{p}_u^\top \mathbf{q}_i)$ whereas our algorithm computes the gradient based on $r_{u,i} - \tilde{\mathbf{p}}_u^\top \tilde{\mathbf{q}}_i$ which can be expressed as $r_{u,i} + n_{u,i} - \mathbf{p}_u^\top \mathbf{q}_i$. This shows the equivalence with working with noisy observed ratings.

This noise is due to the address hash collision among different elements. Thanks to the pairwise independence assumption, the error is independent of the cell location. Also the sign hash function $s_j(\cdot)$ makes sure that the expected error on the components of the latent factor is centered: $\mathbb{E}(\tilde{p}_{u,l} - p_{u,l}) = 0$. This shows that the equivalent noises $n_{u,i}$ on the observations are i.i.d. and centered. We surmise that these indeed provide regularization capabilities. We experimentally prove the claim in section 5.2 by showing that the performance of the our system is less sensitive w.r.t. the regularization parameter λ.

Our algorithm uses regularization with Tikhonov penalization: $\lambda \sum_{(u,i)} \|\tilde{\mathbf{p}}_u\|^2 + \|\tilde{\mathbf{q}}_i\|^2$. This has also an interpretation: the variance of the count sketch estimation error is proportional to σ^2 (Sect. 2.3), which, in our case, is $\sum_{(u,i)} \|\tilde{\mathbf{p}}_u\|^2 + \|\tilde{\mathbf{q}}_i\|^2$. Our method thus aims at minimizing a combination of the error for predicting the ratings and the error for estimating latent vectors from the count sketch.

4.3 Differential privacy

Our algorithm inherently adds noise on the updates thanks to the count sketch. It looks like the SGLD (11) (12), but this is not. We can find the following differences. First, when updated, latent factors related to observed $r_{u,i}$ are not corrupted by noise, whereas all the others are. Second, the noise induced by the count sketch has a variance equalling δ^2/wk where δ is the last update (see Sec. 2.3). Therefore, this variance is proportional to η^2/wk and not η as in the SGLD algorithm. Third, this noise results from $2d$ latent factors updates only therefore it is certainly not Gaussian distributed.

The differential privacy is enforced by clipping the log-likelihood s.t.

$$|\log p(r_{u,i}|\mathbf{P},\mathbf{Q})| = \begin{cases} (r_{u,i} - \hat{r}_{u,i})^2/2 & \text{if } (r_{u,i} - \hat{r}_{u,i})^2 \leq \epsilon \\ \epsilon/2 & \text{otherwise} \end{cases}$$

This enables ϵ-DP as shown in [28, 19], except that this no longer defines a valid conditional probability.

5. EXPERIMENTS

This section evaluates the claimed benefits of our approach. We first experimentally study its regularization capabilities. Then we analyze the privacy-utility trade-off and compare to regular factorization (without privacy).

5.1 Setup

5.1.1 Dataset

We use two publicly available datasets: Movielens1M [1] and EachMovie. Data characteristics are in Table 1. The data is preprocessed and randomly partitioned into the training, validation and test sets with proportion $[0.8, 0.1, 0.1]$. Preprocessing includes bias correction and frequency based thresholding: User, item and global means are substracted from ratings; ratings with user/item frequency < 10 are removed from the test and validation sets.

5.1.2 Evaluation

We use root mean square error to measure the quality of recommendations. The error materializes the deviation of the predicted rating from its true value. This squared error is averaged over the testing set:

$$RMSE(\mathbf{R}') = \sqrt{\frac{1}{\|\mathbf{R}'\|_0} \sum_{r_{u,i} \in \mathbf{R}'} (\tilde{\mathbf{p}}_u^\top \tilde{\mathbf{q}}_i - r_{u,i})^2}, \qquad (18)$$

where $\mathbf{R}'$ is the restriction of $\mathbf{R}$ to the testing set.

We use Kullback-Leibler divergence to gauge privacy. The KL divergence gives the expected amount of information 'leaked' about the fact that user u submitted his rating about item i. This divergence quantifies the difference between the probability distributions of the prediction for observed ratings (the training set) and non observed ratings (the testing set). A low divergence means that the predicted rating $\hat{r}_{u,i}$ is statistically similar whether $r_{u,i}$ was used in training or not. An attacker observing $\hat{r}_{u,i}$ and *even knowing* $r_{u,i}$ can not decide whether this rating was submitted to the recommendation system. To this aim, we experimentally observed that the prediction error $(\hat{r}_{u,i} - r_{u,i}) \sim \mathcal{N}(m, s^2)$. We measure (m_{tr}, s^2_{tr}) over the training set and (m_{te}, s^2_{te}) over the testing set, and compute

$$KLD = \frac{1}{2}\left(\frac{s^2_{tr}}{s^2_{te}} + \frac{(m_{tr} - m_{te})^2}{s^2_{te}} - 1 + \log\frac{s^2_{te}}{s^2_{tr}}\right). \qquad (19)$$

5.1.3 Parameters

We compare the performance for various configurations (w, k) of the sketch and different latent factor dimensions. The sketch depth k is picked from $\{1, 4\}$ and the latent factor dimension d is chosen from $\{8, 16, 32\}$. We measure the space gain γ by the ratio of space that the regular factorization would need for the same dimension d to the space actually utilized by sketch based factorization. We vary γ within $\{1, 2, 4\}$. We determine the sketch width based on the space gain, dimension d and sketch depth k:

$$w = \left\lceil \frac{(|\mathbb{U}| + |\mathbb{I}|)d}{\gamma k} \right\rceil. \qquad (20)$$

We choose optimal parameters for learning rate η and regularization constant λ by a two stage line search in log-scale, based on validation set prediction score. We iterate for $T = 100$ epochs over the training set, before predicting on the testing set, measuring $RMSE(\mathbf{R}')$ and KLD. Learning rate is scaled down using the formula $\eta_t = \frac{\eta}{1+b\cdot t/T}$.

Dataset	$\lvert\mathbb{U}\rvert$	$\lvert\mathbb{I}\rvert$	$\lvert\mathbf{R}\rvert$	rating
MovieLens 1M	6,040	3,952	1,000,209	1:5 (5)
EachMovie	61,265	1,623	2,811,718	1:6 (6)

Table 1: Dataset characteristics

5.2 Regularizing effect of count sketch

We now study the effect of regularization parameter λ on RMSE. We use EachMovie dataset under the setup ($d = 32$, $\gamma = 1$). We vary λ from 10 to 0 in log scale. Figure 1 compares the performance. We benchmark our method ($k = 4$) against regular online matrix factorization and feature hashing based factorization [16] as it is a special case of our approach ($k = 1$). The three techniques share the following

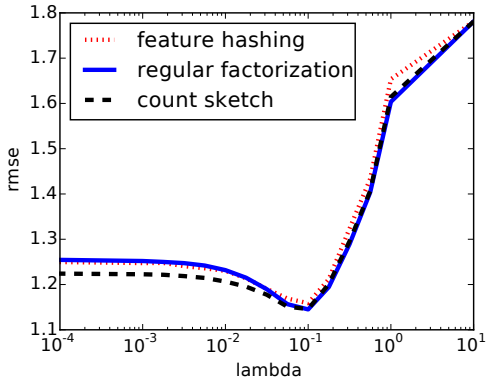

Figure 1: RMSE w.r.t. λ, EachMovie

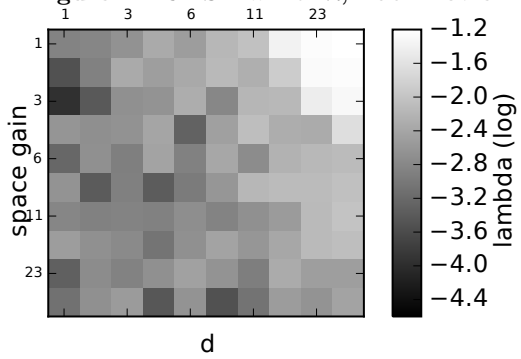

Figure 2: Best λ w.r.t. (d, γ), MovieLens 1M

observations: The optimal λ is around 0.1 and the RMSE increases with λ beyond this value. When we decrease λ below the optimal value, the RMSE degrades and this is attributed to overfitting. The degradation of count sketch factorization is not as worse as the other two techniques. Even when the regularization is turned off ($\lambda = 0$), our scheme performs better than the other two, for the same d. This shows that the 'noisy' sketch structure by itself provides some regularization capabilities.

The heatmap of Figure 2 represents the optimal lambda values for various (d, γ) pairs on MovieLens 1M dataset. The optimal value is often lower than 10^{-2}, except in the top right corner, where γ is small while d is big. This setting indeed ensures superfluous parameter space, which does require stronger regularization to avoid overfitting. The λ value diminishes with d: a smaller model requires less regularization. An interesting observation is that λ lowers with increase in γ and it is true even for a fixed d. This increases address hash collisions and hence the variance of the count sketch estimation (Sect. 4.2) which helps model generalization like when learning on noisy data.

5.3 Privacy utility tradeoff

In this section, we compare the variation of RMSE with respect to parameter ϵ as shown in figure 3. We vary ϵ in the range of $\{2, 4, 8, \infty\}$ and benchmark it against regular factorization. As expected, lowering the privacy by increasing ϵ improves the utility, *i.e.* decreases RMSE. The improvement is significant in the lower range of ϵ and drops gradually as we increase ϵ further. When the epsilon is above 16 the RMSE is close to regular factorization.

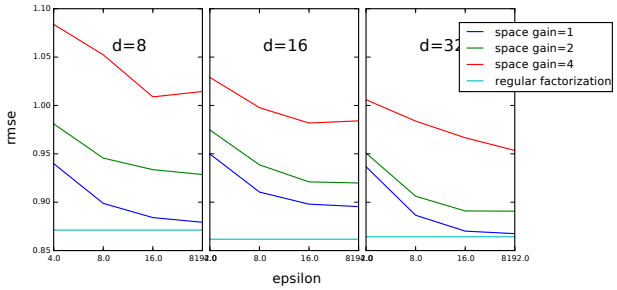

Figure 3: RMSE w.r.t. ϵ, Movielens

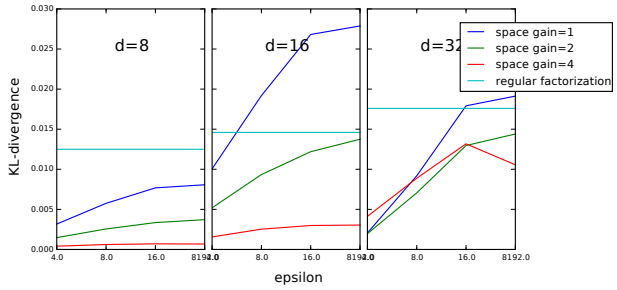

Figure 4: KL divergence w.r.t. ϵ, Movielens

5.4 KL divergence as a privacy measure

We take the role of an attacker willing to know whether a particular rating was used in the training. We use KL divergence to measure the amount of information 'leaked' by getting access to its prediction. Figure 4 shows that KL divergence increases along with the targeted level of privacy ϵ. The KL divergence of regular factorization without any differential privacy mechanism is higher for most of the cases. This supports our approach. It is also clear from the figure that compact models (higher γ or lower d) produce recommendations leaking less information compared to bigger one. The KL divergence which is a measurement on average is indeed much smaller than the target ϵ which is a guaranty on the worst case.

We also had a focused on 'top users', *i.e.* users who submitted a lot. As foreseen, the RMSE measured on this population is lower: they get better predictions because their models are more precise as they are learned on more data. The difference in performance is indeed small. Yet, very surprisingly, the KL divergence measured on this population is smaller by one order of magnitude. This population enjoys better recommendations together with better privacy.

6. RELATED WORK

Many mechanisms enable differential privacy including Laplacian mechanism [9], exponential mechanism [21], Bayesian inference [28], smooth sensitivity, and sample-aggregate frameworks [25]. The works differ on the stage in which the randomization mechanism is added to the system. Initial approaches were advocating at the input and output levels. In [6], they incorporate it at the optimization level and demonstrated its superiority. Differential privacy is also applied to online learning in [13]. A detailed survey of different techniques is available at [26] and [14]. [11] applied differential privacy to data mining. [17] studied the theoretical

properties of learning under differential privacy setup.

Recommender systems is one of the compelling applications of differential privacy owing to its data involving sensitive user information. Many approaches have been proposed in the past for different collaborative filtering approaches like neighbor based approaches and matrix factorization. [23] used sketch to provide privacy. Their notion of privacy is very similar to differential privacy except that the log-likelihood ratio is bounded by a linear parameter instead of exponential. [22] proposed a recommender system by aggregating co-occurrence count in a privacy preserving way. Paper [20] applied differential privacy to the global statistics collected from the rating matrix.

As for matrix factorization, paper [2] compares different ways to ensure differential privacy, among them, the Laplace mechanism on the inputs or on the updates of the stochastic gradient descent. [15] proposed a differentially private low rank approximation using exponential mechanism. [24] devised a matrix factorization using cryptographic garbled circuits, but this does not enable differential privacy.

7. CONCLUSION

This work shows that sketching techniques can be used to preserve privacy by taking advantage of their inherent randomness. This is in contrast to conventional techniques which uses special mechanism to achieve the same. We also get additional benefits like scalability and adaptivity to dynamic data, making it preferable for realistic large-scale applications. We experimentally validate our approach using standard datasets. Future works include measurement of information leaked by the latent factors themselves. This is relevant when the latent vectors are learned by the server and sent to the users allowing local recommendation.

8. REFERENCES

[1] http://grouplens.org/datasets/movielens/.

[2] A. Berlioz, A. Friedman, M. A. Kaafar, R. Boreli, and S. Berkovsky. Applying differential privacy to matrix factorization. In *RecSys*, RecSys '15, pages 107–114, New York, NY, USA, 2015. ACM.

[3] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[4] J. Calandrino, A. Kilzer, A. Narayanan, E. Felten, and V. Shmatikov. "you might also like:" privacy risks of collaborative filtering. In *SP*, pages 231–246, May 2011.

[5] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *ICALP*. 2002.

[6] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12:1069–1109, July 2011.

[7] G. Cormode. Sketch techniques for approximate query processing. *FnTD. NOW publishers*, 2011.

[8] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[9] C. Dwork. *ICALP 2006*, chapter Differential Privacy, pages 1–12. Berlin, Heidelberg, 2006.

[10] C. Dwork. *Encyclopedia of Cryptography and Security*, chapter Differential Privacy, pages 338–340. Springer US, Boston, MA, 2011.

[11] A. Friedman and A. Schuster. Data mining with differential privacy. In *SIGKDD*, KDD '10, pages 493–502, New York, NY, USA, 2010. ACM.

[12] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, pages 604–613. ACM, 1998.

[13] P. Jain, P. Kothari, and A. Thakurta. Differentially private online learning. 09 2011.

[14] Z. Ji, Z. C. Lipton, and C. Elkan. Differential privacy and machine learning: a survey and review. *CoRR*, abs/1412.7584, 2014.

[15] M. Kapralov and K. Talwar. On differentially private low rank approximation. In *ACM-SIAM*, pages 1395–1414. SIAM, 2013.

[16] A. Karatzoglou, M. Weimer, and A. J. Smola. Collaborative filtering on a budget. In *AISTATS*, 2010.

[17] S. P. Kasiviswanathan, H. K. Lee, K. Nissim, S. Raskhodnikova, and A. Smith. What can we learn privately? *SIAM Journal on Computing*, 40(3):793–826, 2011.

[18] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[19] Z. Liu, Y.-X. Wang, and A. Smola. Fast differentially private matrix factorization. In *RecSys*, RecSys '15, pages 171–178, New York, NY, USA, 2015. ACM.

[20] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the net. In *SIGKDD*, KDD '09, pages 627–636, New York, NY, USA, 2009. ACM.

[21] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103, Oct 2007.

[22] L. Melis, G. Danezis, and E. D. Cristofaro. Efficient private statistics with succinct sketches. *CoRR*, abs/1508.06110, 2015.

[23] N. Mishra and M. Sandler. Privacy via pseudorandom sketches. In *ACM SIGMOD-SIGACT-SIGART*, pages 143–152. ACM, 2006.

[24] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh. Privacy-preserving matrix factorization. In *SIGSAC*, pages 801–812. ACM, 2013.

[25] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM.

[26] A. Sarwate and K. Chaudhuri. Signal processing and machine learning with differential privacy: Algorithms and challenges for continuous data. *Signal Processing Magazine, IEEE*, 30(5):86–94, Sept 2013.

[27] S. J. Vollmer, K. C. Zygalakis, and Y. Teh. (non-) asymptotic properties of stochastic gradient langevin dynamics. *arXiv:1501.00438*, 2015.

[28] Y.-X. Wang, S. E. Fienberg, and A. Smola. Privacy for free: Posterior sampling and stochastic gradient monte carlo. *arXiv preprint arXiv:1502.07645*, 2015.

[29] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, pages 681–688, Bellevue, WA, USA, 2011.

[30] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *arXiv preprint arXiv:1411.4357*, 2014.